

When Can Helper Node Selection Improve Regenerating Codes? Part II: An Explicit Exact-Repair Code Construction

Imad Ahmad, *Student Member, IEEE*, and Chih-Chun Wang, *Senior Member, IEEE*

Abstract—Part I of this work answered the following fundamental question: Under and only under what (n, k, d) values does proactively choosing the helper nodes improve the storage-bandwidth tradeoff of regenerating codes (RCs)? A critical component of the achievability results in Part I is a low-complexity helper selection solution, termed the *family helper selection (FHS)* scheme. In Part I, FHS was proved to be *optimal* or *weakly optimal* (depending on the underlying scenario) based on the assumption that as long as the *minimum min-cut value of FHS is no less than the file size*, we can always find a distributed storage code that can reconstruct the original file. This work relaxes this assumption by providing an exact-repair code construction that attains the minimum-bandwidth-regenerating (MBR) point of FHS, previously calculated/predicted only by the min-cut analysis in Part I. As an outcome, for the MBR points, FHS is indeed optimal or weakly optimal under various scenarios.

In addition to closing the loop of the graph-based study in Part I, the proposed construction can also be viewed as a generalization of the existing fractional repetition (FR) codes. FR codes are exact-repair codes that admit the highly-desirable *repair-by-transfer* property. However, the unique construction of FR codes limits its application to a restricted set of (n, k, d) system parameters. In contrast, our new construction, termed the generalized FR (GFR) codes, can be applied to arbitrary (n, k, d) values. Our GFR codes retain most of the practical benefits of FR codes, i.e., being exact-repair and being *almost* repairable-by-transfer.

Index Terms—Distributed storage, regenerating codes, family helper selection schemes, helper nodes, generalized fractional repetition codes, network coding

I. INTRODUCTION

IN Part I [2] of this paper, we provided a necessary and sufficient condition for the (n, k, d) system parameters under which (carefully) choosing the helpers during repair improves the storage-bandwidth tradeoff of regenerating codes (RCs). To prove the sufficiency of the condition, the achievability part, we devised a helper selection scheme termed the *family helper selection (FHS)* scheme and we characterized its storage-bandwidth tradeoff. Specifically, for any given (n, k, d) parameters, the tradeoff curve of the FHS scheme was derived using a graph-based analysis by quantifying the minimum possible min-cut value of FHS.

In the original work on RCs [4], it was assumed that there always exists network codes that can achieve the min-cut-based tradeoff curve of RCs with blind helper selection

(BHS) [1], [10]. Unfortunately, such an assumption needs to be rigorously proved since the results in [1], [10] only guarantee the existence of network codes¹ for any *network/graph of bounded size*, but the size of the information flow graph (IFG) of a distributed storage code (see [4] or Part I [2] for the description of these graphs) is unbounded due to the nature of the repair problem. For that reason, a follow-up work [18] rigorously proved the assumption true. Specifically, it outlined a detailed code construction that achieves all the points on the storage-bandwidth tradeoff curve of the original RCs [4], which was previously computed/predicted by a pure min-cut based analysis.

The codes provided in [18] are *functional-repair* codes, i.e., they allow a newcomer node to store a function of the file that is different from the data that was originally present on the failed node. For practical reasons, a big amount of subsequent development in this direction has been devoted for the study and construction of RCs that exhibit the *exact-repair* property [3], [5], [14], [16], [17], [19], [20]: during repair, the newcomer has to restore the data that was originally stored on the failed node. Exact-repair code construction that achieves the minimum-bandwidth-regenerating (MBR) point of RCs with BHS were proposed in [14]. The construction of those codes is based on product-matrix construction and works for all system parameters (n, k, d) . Different than the product-matrix construction, [5] proposed table-based exact-repair codes called *fractional repetition (FR) codes* for the scenario in which the (n, k, d) values satisfy $n \cdot d$ being even. For those (n, k, d) values, FR codes can again achieve and sometimes surpass the MBR point of BHS, see [5].

Although the product-matrix codes and the FR codes are both exact-repair and can achieve the MBR point of BHS, each type of codes has a distinct advantage over the other. For example, the product-matrix codes [14] can naturally *handle multiple failures* since it guarantees that the newcomer can repair from *any* set of d surviving nodes. In contrast, the FR codes rely on the concept of *repetition* and thus were originally designed for the single-failure scenario. There are new generalizations of FR codes for multiple failures [5], [9], [11], [12], but they are at the cost of decreasing the performance (not necessarily achieving the MBR point of BHS anymore) and further restricting the applicable (n, k, d) values. On the other hand, FR codes exhibit a very practically appealing property,

This work was supported in parts by NSF grants CCF-0845968, CNS-0905331, and CCF-1422997.

I. Ahmad and C.-C. Wang are with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, 47906 USA e-mail: {ahmadi, chihw}@purdue.edu.

¹The underlying finite field size may grow with respect to the size of the graph, which is why a finite-field code may not exist for a graph that continuously grows to infinity.

the *repair-by-transfer* property (also referred to as the *uncoded repair* property). Specifically, FR codes only require that helpers send a subset of the packets they store (without mixing them) to the newcomer and hence the name repair-by-transfer. In product-matrix-based codes, during repair, the helpers need to mix their data before sending it to the newcomer in order to repair the failed node. In addition to their practical values, both the product-matrix codes and the FR codes are theoretically important as their existence proves that the MBR point of the RCs with BHS (previously computed/predicted by min-cut analysis) can indeed be achieved by exact-repair codes.

Paralleling the original min-cut-based results in the literature on RCs with BHS [4], Part I of this work focused on analyzing the min-cut values of RCs when different types of helper selection schemes are used. Similar to [4], such a min-cut-based study allows us to characterize the performance of RCs in a closed form without delving into explicit but rather technical code constructions, cf. [5], [14], [18]. In Part II of this work, we turn our focus to explicit code design and propose a new class of codes, termed the *generalized fractional repetition (GFR) codes*, that attain the performance calculated/predicted by the min-cut analysis in Part I.

The contribution of this new class of codes is 3-fold: (1) it closes the loop of the min-cut-based analysis in Part I by explicitly designing a code that achieves the MBR point of the FHS scheme. As a result, the *optimality* and *weak optimality* results of FHS, which were previously proved only through a min-cut-based analysis in Part I, can indeed be realized through explicit GFR code construction when focusing on the MBR point. Since the benefits of helper selection are greatest at the MBR point, the GFR codes we provide in this paper, along with the results of Part I, complete our mission of understanding under what condition can helper selection improve the performance of regenerating codes.

(2) The proposed GFR construction can also be viewed as a generalization of the existing FR codes. However, unlike FR codes which require that $n \cdot d$ be even, our GFR codes are applicable to arbitrary (n, k, d) values while retaining most of the practical benefits of FR codes, i.e., GFR codes are exact-repair codes and the majority of the nodes of a GFR code can be *repaired-by-transfer*.

(3) The analysis of FR codes relies on the edge-counting arguments of the underlying regular “repair-by-transfer graph.” The repair-by-transfer graph being regular is one of the key enablers of the FR analysis, which, unfortunately, is also the reason why FR codes can only be applied when $n \cdot d$ is even. In contrast, for any arbitrary (n, k, d) values, the analysis of GFR codes with FHS relies on (i) new edge-counting arguments for *irregular* repair graphs and on (ii) establishing the connection between the edge counts of the *repair graph* and the min-cut values of the *information flow graph*. By proving/quantifying the benefits of GFR codes for arbitrary (n, k, d) values, such a new analysis approach will further enrich the literature of FR codes and broaden their applications.

The rest of this paper is organized as follows. Section II reviews existing work on FR codes. Section III gives the notation used in this paper and reminds the reader of the necessary and sufficient condition and FHS of Part I. Sec-

tion IV motivates GFR codes by providing three examples that demonstrate their construction. Section V presents the construction of GFR codes. Section VI is concerned with the analysis of GFR codes. Section VII extends GFR codes to the family-plus helper selection scheme of Part I. Section VIII concludes this paper.

II. RELATED WORK

The first construction of a special-case FR code appeared in [15], [16] (although not termed FR initially). The code construction in there was based on encoding the file first using an MDS code and then assigning the encoded packets to the edges of a complete graph. This code construction achieves the MBR point of RCs with $d = n - 1$. Shortly after, the concept of assigning MDS-coded packets to edges of graphs was generalized to hypergraphs and the term “FR codes” was coined in [5], in which the MDS code was referred to as the *outer code* and the repetition code as the *inner code*. These codes include the previous code construction in [15], [16] as special cases.

In FR codes, the number of times packets are replicated in the network is termed the repetition degree. FR codes with repetition degree 2 were proposed in [5] based on regular graphs and are shown to achieve the MBR point of RCs with BHS if and only if $n \cdot d$ is even. Utilizing Steiner systems, [5] was able to construct FR codes for when the repetition degree is larger than 2. The higher the repetition degree the more robust the FR code is since a repetition of $r \geq 2$ times means that the helper of a newcomer can be one of any $r - 1$ other nodes that store the same copy of the packet.

A subsequent work [13] proposed DRESS codes that are randomly constructed codes utilizing the same code construction idea of FR codes. Reference [9] presented graph-based constructions of Steiner systems that translate into FR codes. Those constructions are for the case when the repetition degree is much smaller than the storage allowed per node. FR code constructions using *resolvable designs* were given in [11] that are able to cover a set of parameters not covered by Steiner systems. Moreover, [21] gave FR constructions for storage networks with heterogeneous numbers of helpers. On a different note, FR codes were used as local codes in [7], and [12] gave FR code constructions with local repair, specifically, with $d < k$ using the RCs notation.

All FR codes in the above mentioned works are based on the following two steps. Step 1: Construct MDS coded packets and duplicate each of them $r \geq 2$ times; and Step 2: the duplicated copies are carefully distributed and stored in different network nodes. Those nodes that store the same packet will be helpers of each other when one of the nodes fails. Due to such a 2-step process, all the design and analysis efforts of FR codes are focused on Step 2: how to disseminate the duplicated packets or, equivalently, who will be the helpers when a node fails. Once the design of Step 2 is finalized, we can then analyze the performance of the FR code by an *edge-counting* argument. Although the 2-step process is very powerful, only for a restricted collection of (n, k, d) values can we successfully complete Steps 1 and 2, which thus limits the application of FR codes.

In contrast, the main contribution of the proposed GFR codes is not about designing Step 2. Instead, the helper selection of GFR codes are designed by the graph-based min-cut analysis in Part I of this work [2]. Specifically, Part I shows that in terms of the *min-cut values*, a new helper selection policy, called family helper selection (FHS), is guaranteed to be *optimal* for some (n, k, d) values and *weakly optimal* in general. GFR codes thus directly use the FHS scheme in its Step 2 without any modification. However, it turns out that, with FHS in Step 2, it becomes impossible to reuse the original Step 1. Instead, the focus of the GFR codes is to modify Step 1 so that the combination of *the modified Step 1* and *the use of the FHS scheme in Step 2* results in a code that realizes the optimality and/or weak optimality promised by the min-cut analysis in Part I of this work. By jointly revising Steps 1 and 2, the proposed GFR code can be applied to any (n, k, d) values while retaining most of the practical appeals of the original FR codes, e.g., exact-repair and repair-by-transfer. This was previously not possible when the design efforts were focused on Step 2 only.

III. FLASHBACK OF PART I AND NOTATION

Using the same notation of Part I, we denote the total number of nodes in a network by n . The number of helper nodes, the nodes participating in the repair of a failed node, is denoted by d . This means that during repair, the node that is replacing the failed node, called the newcomer, can contact d nodes for repair. For the reliability requirement, we require that any set of k nodes of the total n nodes be able to reconstruct the original data (for a detailed explanation of the parameters d and k and the distinction between the *desired protection level* k and the *actual achievable protection level* k^* of a code, see Part I [2]). By the nature of the repair problem, (n, k, d) have to satisfy

$$2 \leq n, \quad 1 \leq k \leq n-1, \quad \text{and} \quad 1 \leq d \leq n-1. \quad (1)$$

Furthermore, we denote the amount of storage-per-node by α and the amount of communications or bandwidth-per-helper by β . The size of the original data (also referred to as the original file) is denoted by \mathcal{M} . Similar to the original RC work [4], the min-cut analysis of Part I of this work focuses on the concept of information flow graphs (IFGs). For a detailed description of the IFG, we refer the reader to Part I [2] or [4].

In the next section, we introduce some of the main results of Part I [2] that will be needed in this paper.

A. The Necessary and Sufficient Condition

We studied in Part I the problem of helper selection in RCs. Specifically, Part I characterized the collection (n, k, d) for which we will see strict storage-bandwidth tradeoff improvement if we let the newcomer proactively choose its helpers.

Proposition 1 (Proposition 1 in [2]): If at least one of the following two conditions is true (i) $d = 1$, $k = 3$, and n is odd; and (ii) $k \leq \left\lceil \frac{n}{n-d} \right\rceil$, then choosing the helpers cannot improve the storage-bandwidth tradeoff of RCs with BHS. That is, even the best helper selection scheme is no better than the original

BHS scheme in [4]. Conversely, for any (n, k, d) values that satisfy neither (i) nor (ii), there exists a helper selection scheme such that its storage-bandwidth tradeoff is strictly better than BHS.

The achievability part of the above results, i.e., the existence of a helper selection scheme that can do strictly better, is achieved by analyzing the min-cut values of a new class of helper selection schemes termed *the family helper selection (FHS) scheme*. The basic concepts and notation of the FHS scheme will be introduced in the next subsection.

B. The Family Helper Selection Scheme

In Part I, we classified helper selection schemes under three main types:

- *Dynamic helper selection (DHS)*: the helper node choice can depend on the history of all the previous time slots 1 to $(\tau - 1)$. Mathematically, the helper set decision at time τ can be written in function form as $D_\tau(\{F_l\}_{l=1}^{\tau-1})$ that returns the set of helpers the newcomer has to access at time τ , where F_l is the failed node at time l .
- *Stationary helper selection (SHS)*: a subset of the DHS schemes that assigns fixed helper sets of d nodes to each node. Mathematically, each node index i is associated with a set of indices D_i where the size of D_i is d . Whenever node i fails, the newcomer (for node i) simply accesses those helpers j in D_i . Equivalently, a SHS scheme can be expressed by a function $D(F_\tau) = D_i$ if $F_\tau = i$ (i.e., the i -th node has failed in time τ). Comparing SHS with the most general DHS schemes, the main difference is that the function $D(\cdot)$ no longer depends on τ nor the history of the failure pattern $\{F_l\}_{l=1}^{\tau-1}$, and it depends only on the current failed node index F_τ .
- *Blind helper selection*: as assumed in [4], the helpers are chosen arbitrarily at all times of repair.

To prove the sufficiency of the condition found in Part I, a new low-complexity helper selection scheme, termed *the family helper selection (FHS) scheme*, was proposed which belongs to the class of SHS schemes and thus can be described by $\{D_i : i = 1, \dots, n\}$. The FHS scheme can be applied to any arbitrary (n, k, d) values in the following way. The storage nodes are denoted by 1 to n . Then, the first $(n-d)$ nodes are grouped as the first *complete family* and the second $(n-d)$ nodes are grouped as the second complete family and so on. In total, there are $\left\lfloor \frac{n}{n-d} \right\rfloor$ complete families. The remaining $n \bmod (n-d)$ nodes are grouped as an *incomplete family*. The helper set D_i of any node i in a complete family contains all the nodes *not* in the same family of node i . For any node in the incomplete family, we set the corresponding $D_i = \{1, \dots, d\}$. An example of the FHS scheme will be provided in Section IV-C. The FHS scheme is the building foundation of the achievability analysis of Proposition 1.

IV. THREE EXAMPLES THAT DEMONSTRATE THE CONSTRUCTION OF GFR CODES

In this section, we give three examples to highlight the main ideas of the proposed GFR codes.

A. Example 1: The FR Codes Can be Very Powerful

In the following example, we demonstrate how to construct an FR code for the parameter value $(n, k, d, \alpha, \beta) = (5, 3, 2, 2, 1)$.

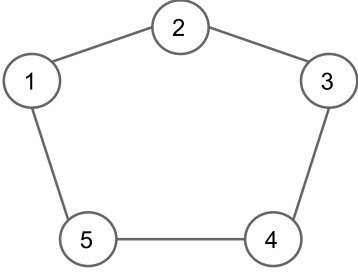


Fig. 1. The regular graph of the FR code for $(n, k, d, \alpha, \beta) = (5, 3, 2, 2, 1)$.

For this set of parameters, we know that we can construct an FR code [5] since $n \cdot d$ is even. As described in [5], we have to first construct a regular graph with n nodes and each node has degree d , representing the inner code, which can be generated randomly and efficiently, e.g., using the algorithm in [8]. For $(n, d) = (5, 2)$, there exists only one possible regular graph with $n = 5$ nodes/vertices and degree $d = 2$, which is shown in Fig. 1. This regular graph has $\frac{nd}{2} = 5$ edges. Observe that if we pick any set of $k = 3$ nodes of this graph, the number of distinct edges incident to this set is 4.

This means that we can protect a file of $\mathcal{M} = 4$ packets as follows. Encode a file of 4 packets using a $(5, 4)$ -MDS code (the outer code), a simple binary code with a parity-bit would do. Then assign the 5 MDS-coded packets to the five edges of the regular graph, one for each edge. Each physical node stores the packets of the edges incident to the vertex corresponding to it.

Repair: Following the graph in Fig. 1, if a node fails, then it simply contacts its $d = 2$ neighboring nodes in the graph and downloads its packets from them [5]. For example, if storage node 4 fails, then it contacts nodes 3 and 5 and downloads its packets.

Reconstruction: If we pick any set S of $k = 3$ storage nodes of the total $n = 5$ storage nodes in the network, then we can get 4 distinct packets of the MDS code. Since it is an MDS code, we can reconstruct the original file.

The largest file size that can be protected by the original RCs with BHS [4] with $(n, k, d, \alpha, \beta) = (5, 3, 2, 2, 1)$ is $\mathcal{M} = 3$ packets. If we plug in the $(n, k, d) = (5, 3, 2)$ into Proposition 1, we have that $k = 3 > \left\lceil \frac{n}{n-d} \right\rceil = \left\lceil \frac{5}{3} \right\rceil = 2$, which implies that a carefully devised helper selection scheme can strictly outperform RCs with BHS. Since FR codes can be viewed as a special example of SHS schemes, the FR code in this example (see Fig. 1) concurs with the prediction of Proposition 1. By some simple min-cut-based analysis, one can actually prove that the FR code is *absolutely optimal* for this (n, k, d, α, β) values. Specifically, it is impossible to design any code that satisfies $(n, k, d, \alpha, \beta) = (5, 3, 2, 2, 1)$ and can protect a file size \mathcal{M} strictly larger than 4.

This example demonstrates the power of FR codes, and shows that for some instances, FR codes can be optimal while

admitting very desirable features, e.g., repair-by-transfer.

B. Example 2: Not All FR codes Are Equal

Consider the parameter value $(n, k, d, \alpha, \beta) = (6, 3, 3, 3, 1)$. As described in [5], the FR code is based on constructing a regular graph of $n = 6$ nodes and node degree $d = 3$. This is possible since $n \cdot d$ is even. A possible regular graph for this example is shown in Fig. 2.

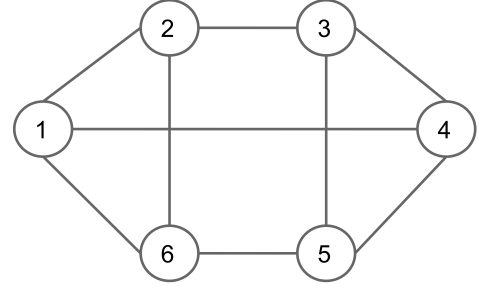


Fig. 2. The regular graph of the FR code for $(n, k, d, \alpha, \beta) = (6, 3, 3, 3, 1)$.

Using the regular graph in Fig. 2, we can construct an FR code that can protect $\mathcal{M} = 6$ packets. The construction is as follows. First use a $(9, 6)$ -MDS code to convert the 6 original packets into 9 MDS-coded packets. Then, each MDS coded packet is assigned to one of the 9 edges in Fig. 2. Each node will then store the $d = 3$ packets corresponding to its 3 adjacent edges. To see that any $k = 3$ nodes can reconstruct the original file, we observe that any 3 nodes have ≥ 6 distinct edges incident to them. E.g., nodes $\{1, 2, 6\}$ have exactly 6 adjacent edges. Then, by the MDS property, these ≥ 6 MDS-coded packets can be used to reconstruct the original file. By [4], RCs with BHS can also protect $\mathcal{M} = 6$. The above FR construction can thus be viewed as an achievability scheme for RCs with BHS.

The remaining question to answer is whether we can design an FR code that can protect $\mathcal{M} > 6$ packets. By plugging in the $(n, k, d) = (6, 3, 3)$ value into Proposition 1, we have $k = 3 > \left\lceil \frac{n}{n-d} \right\rceil = \left\lceil \frac{6}{3} \right\rceil = 2$, which implies that there indeed exists a scheme that outperforms RCs with BHS [4]. In the following, we describe another FR code that achieves $\mathcal{M} = 7$.

We observe that the regular graph for this example is actually not unique. Instead, we can consider another regular graph in Fig 3 which also has $n = 6$ nodes and node degree $d = 3$. We observe that in this new regular graph, any $k = 3$ nodes have ≥ 7 distinct adjacent edges. As a result, if we use a $(9, 7)$ -MDS code in the beginning and use the regular graph in Fig. 3, then the resulting FR code can protect a file of size $\mathcal{M} = 7$.

This example demonstrates that the performance of an FR code depends on how one chooses the underlying regular graph. The FR code based on Fig. 3 is strictly better than the FR code based on Fig. 2. As discussed in Section II, almost all existing works on FR codes focus on designing the underlying regular graph.

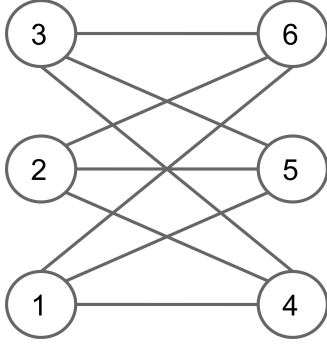


Fig. 3. An alternative regular graph for $(n, k, d, \alpha, \beta) = (6, 3, 3, 3, 1)$.

C. Example 3: Sometimes No FR is Good Enough

We use the parameter value $(n, k, d, \alpha, \beta) = (7, 3, 3, 3, 1)$ to demonstrate the limitation of FR codes and how our GFR codes work. For $(n, k, d, \alpha, \beta) = (7, 3, 3, 3, 1)$, RCs with BHS in [4] can protect a file of size $\mathcal{M} = 6$. However, by plugging in Proposition 1, we have $k = 3 > \left\lceil \frac{n}{n-d} \right\rceil = \left\lceil \frac{7}{4} \right\rceil = 2$, which implies that there exists a scheme that can protect $\mathcal{M} > 6$ packets, say protect $\mathcal{M} = 7$ packets. The remaining question is how to design such a scheme.

Based on the previous examples, an obvious candidate is to see whether one can design an FR code that achieves $\mathcal{M} = 7$. As mentioned in [5], whether there exists an n -node regular graph with degree d depends on whether $n \cdot d$ is even. For this $(n, k, d) = (7, 3, 3)$, it is thus impossible to construct a regular graph with $n = 7$ nodes and degree $d = 3$, therefore, no FR code exists for $(n, k, d) = (7, 3, 3)$. In contrast, our GFR code will use the FHS scheme in Section III-B and [2] to design the helper node sets when a node fails and can be applied to arbitrary (n, k, d) values.

First, we have that $\left\lceil \frac{n}{n-d} \right\rceil = \left\lceil \frac{7}{4} \right\rceil = 2$. Following the description in Section III-B, we have 1 complete family, nodes $\{1, 2, 3, 4\}$, and 1 incomplete family, nodes $\{5, 6, 7\}$. More specifically, any of nodes 1 to 4 will request help from nodes 5 to 7. Any of nodes 5 to 7 will request help from nodes 1 to 3. Note the asymmetry of the helper relationship, i.e., node 4 requests help from nodes 5 to 7 but is never a helper for nodes 5 to 7. See Fig. 4 for illustration, in which we use the dashed line to represent the asymmetric helper relationship of node 4.

Our GFR code is based on GF(32) and can protect a file of 7 packets while satisfying $(n, k, d, \alpha, \beta) = (7, 3, 3, 3, 1)$. The 7 packets of the file are denoted by W_1, W_2, \dots, W_7 . We first encode the 7 packets into 9 packets X_1 to X_9 where $X_i = W_i$ for $i = 1$ to 7 and X_8 and X_9 are

$$X_8 = 23W_1 + 3W_2 + 9W_3 + 24W_4 + 30W_5 + 8W_6 + 8W_7, \quad (2)$$

$$X_9 = 25W_1 + 25W_2 + 2W_3 + 18W_4 + 12W_5 + 25W_6 + 27W_7. \quad (3)$$

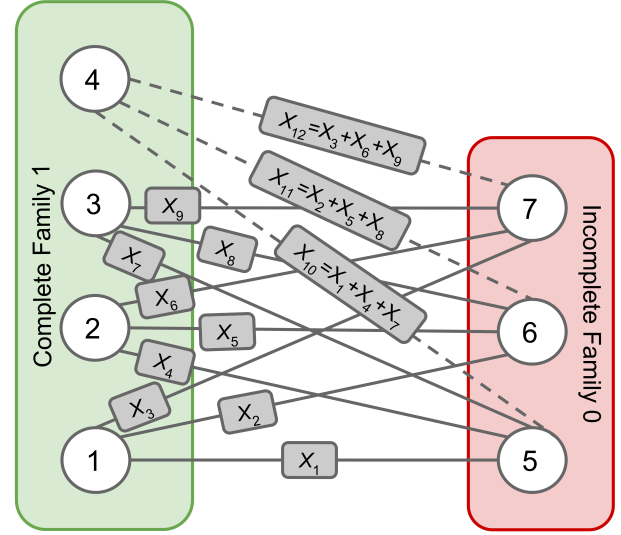


Fig. 4. The graph representation of the code for $(n, k, d, \alpha, \beta) = (7, 3, 3, 3, 1)$.

Finally, we create 3 additional packets X_{10} , X_{11} , and X_{12} by

$$X_{10} = X_1 + X_4 + X_7, \quad (4)$$

$$X_{11} = X_2 + X_5 + X_8, \quad (5)$$

$$X_{12} = X_3 + X_6 + X_9. \quad (6)$$

Once the X_1 to X_{12} packets are encoded from W_1 to W_7 , we assign the packets X_1, X_2, \dots, X_9 to the solid edges as shown in Fig. 4 and assign the packets X_{10}, X_{11} , and X_{12} to the dashed edges incident to incomplete family nodes 5, 6, and 7, respectively. Each physical node in $\{1, 2, 3, 5, 6, 7\}$ (excluding node 4) now stores the packets corresponding to the *solid* edges adjacent to it. Node 4 stores the packets corresponding to the dashed edges incident to itself.² One can clearly see that, in this code construction, each node stores exactly $\alpha = 3$ packets.

Repair: If any of the nodes in $\{1, 2, 3, 5, 6, 7\}$ (excluding node 4) fails, then the newcomer downloads the lost packets of the solid edges from its adjacent nodes. If node 4 fails, then nodes 5, 6, and 7 generate and send to the newcomer the linear combinations $X_1 + X_4 + X_7$, $X_2 + X_5 + X_8$, and $X_3 + X_6 + X_9$, respectively. This is always possible since node 5 stores $\{X_1, X_4, X_7\}$, node 6 stores $\{X_2, X_5, X_8\}$, and node 7 stores $\{X_3, X_6, X_9\}$. Notice that these generated packets correspond to the packets X_{10}, X_{11} , and X_{12} of the dashed edges and node 4 is thus exactly-repaired. Our GFR construction is *almost repairable-by-transfer*, since all nodes but node 4 can be repaired by transfer.

Reconstruction: One can verify, by a computer-based exhaustive search, that the given code assignment can reconstruct the $\mathcal{M} = 7$ packets of the original file from any $k = 3$ nodes of the total $n = 7$ nodes. Note that to verify $\mathcal{M} = 7$, one cannot use the original edge/packet counting arguments in [5] since the underlying graph, see Fig. 4, is non-regular and of asymmetric helper relationship (solid vs. dashed edges). One

²The GFR code construction for general (n, k, d) values will be detailed in Section V-B.

main contribution of this work is to analytically characterize the protected file size \mathcal{M} of our GFR codes for arbitrary (n, k, d) values. Additionally, our results show that the \mathcal{M} value of the GFR code matches the MBR point predicted by the min-cut analysis of the FHS scheme in Part I [2] for any (n, k, d) values, which thus close the loop of the graph-based analysis of Part I.

V. THE GENERALIZED FRACTIONAL REPETITION CODES

The motivation of the GFR design is to achieve the MBR point of the FHS scheme computed by the min-cut analysis in Part I [2]. In this section, we will first describe the MBR point of the FHS scheme and then describe the GFR construction that attains it.

A. The MBR Point of The FHS Scheme

To describe the MBR point of the FHS scheme, we first reintroduce (was introduced in Part I [2]) the notion of the *family index vector* of nodes 1 to n , which is n -dimensional and is defined as

$$\left(\underbrace{1, \dots, 1}_{n-d}, \underbrace{2, \dots, 2}_{n-d}, \dots, \underbrace{c, \dots, c}_{n \bmod (n-d)}, \underbrace{-c, \dots, -c}_{n-d-(n \bmod (n-d))}, \underbrace{0, \dots, 0}_{n \bmod (n-d)} \right), \quad (7)$$

where the idea is that complete families are denoted by 1 to $c \triangleq \lfloor \frac{n}{n-d} \rfloor$. Since any node from the incomplete family only requests help from nodes 1 to d , not all nodes in the last family, family c , will serve as the helpers of the incomplete family members. As a result, the last $(n-d-(n \bmod (n-d)))$ nodes of family c are indexed by $-c$ to distinguish them from the other nodes in the same family. The nodes of the incomplete family are denoted by 0. For any node i , we use $FI(i)$ to denote the family index of i .

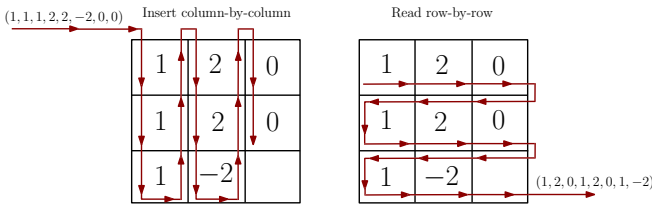


Fig. 5. The construction of the RFIP for $(n, d) = (8, 5)$.

A *family index permutation* is a permutation of the family index vector in (7), which is denoted by π_f . A *rotating family index permutation (RFIP)* π_f^* is a special family index permutation that puts the family indices of (7) in an $(n-d) \times \lfloor \frac{n}{n-d} \rfloor$ table column-by-column and then reads it row-by-row. Fig. 5 illustrates the construction of the RFIP for the case of $(n, d) = (8, 5)$. The input is the family index vector $(1, 1, 1, 2, 2, -2, 0, 0)$ and the output is the RFIP $\pi_f^* = (1, 2, 0, 1, 2, 0, 1, -2)$.

The graph-based analysis in Part I shows that for any given (n, k, d, α, β) values satisfying $\alpha = d\beta$, thus the MBR point, the largest file size \mathcal{M} that can be protected by the FHS scheme is

$$\mathcal{M} = \sum_{i=1}^k (d - y_i(\pi_f^*)) \beta, \quad (8)$$

where the function $y_i(\pi_f^*)$ is computed as follows. If the i -th coordinate of π_f^* is 0, then $y_i(\pi_f^*)$ returns the number of j satisfying both (i) $j < i$ and (ii) the j -th coordinate > 0 . If the i -th coordinate of π_f^* is not 0, then $y_i(\pi_f^*)$ returns the number of j satisfying both (i) $j < i$ and (ii) the absolute value of the j -th coordinate of π_f^* and the absolute value of the i -th coordinate of π_f^* are different. For example, since $\pi_f^* = (1, 2, 0, 1, 2, 0, 1, -2)$, then $y_6(\pi_f^*) = 4$ and $y_8(\pi_f^*) = 5$. Our GFR code is guaranteed to achieve the MBR point in (8).

B. The Construction of GFR Codes

Before describing the construction of GFR codes, we list some notational definitions. We denote the set of nodes of complete family i by N_i . For the last complete family, i.e., $i = c$ where $c = \lfloor \frac{n}{n-d} \rfloor$, we split its nodes into two disjoint node sets, N_{-c} is the set of nodes in family c that is not in the helper set of the incomplete family nodes and N_c is the set of the remaining nodes of this complete family. We denote the set of nodes in the incomplete family by N_0 . The set of all nodes in the network is denoted by N . For example, if $(n, d) = (7, 3)$ as in the example of Section IV-C, we have $N_1 = \{1, 2, 3\}$, $N_{-1} = \{4\}$, and $N_0 = \{5, 6, 7\}$. In short, N_x contains the nodes that have family index x .

We assume without loss of generality that $\beta = 1$ and $\alpha = d$, i.e., one packet is communicated per helper and d packets are stored in each node. By the MBR point formula of the FHS scheme (8), the goal of GFR codes is to protect a file of size

$$\mathcal{M} = \sum_{i=1}^k (d - y_i(\pi_f^*)) \text{ packets} \quad (9)$$

against any $(n - k)$ simultaneous failures. From (9), we can easily see that the larger the k value, the more relaxed the reliability requirement is, and the larger the file size \mathcal{M} the GFR code can protect.

The core idea of GFR codes stems from the concatenation of an inner code that is based on a graph representation of the distributed storage network and a carefully designed outer code that satisfies special properties. We first introduce the graph-based inner code of the GFR code.

The inner code: The inner code is based on the following graph representation of the distributed storage network. Each physical node in the network is represented by a vertex in the graph, which is denoted by $G = (V, E)$ where V denotes the set of vertices of G and E denotes its set of edges. As will be described, the graph consists of two disjoint groups of edges. Graph G has the following properties:

- 1) $V = \{1, 2, \dots, n\}$. Each vertex i in V corresponds to physical node i in N . For convenience, throughout our

discussion, we simply say vertex $i \in N_x$ if the physical node that vertex i corresponds to is in N_x .

- 2) Two vertices $i \in N_x$ and $j \in N_y$ are connected by an edge in E if $|x| \neq |y|$ and $(x, y) \notin \{(0, -c), (-c, 0)\}$. The collection of all those edges is denoted by \bar{E} .
- 3) Two vertices $i \in N_0$ and $j \in N_{-c}$ are connected by an edge in E . The collection of all those edges is denoted by \tilde{E} .
- 4) From the above construction, we have $E = \bar{E} \cup \tilde{E}$. We further assume that all the edges are undirected and there are no parallel edges in G .

Fig. 4 of Example 3 in Section IV is an example of the above graph representation of the inner code. Notice that the edges in \bar{E} are represented by solid lines, while the edges in \tilde{E} are represented by dashed lines.

Recall that $FI(i)$ denotes the family index of node i . We define the following three sets:

$$\begin{aligned} \text{IJ}^{[1]} &= \{(i, j) : 1 \leq i < j \leq n, 1 \leq |FI(i)| < |FI(j)| \leq c\} \\ \text{IJ}^{[2]} &= \{(i, j) : 1 \leq i < j \leq n, 1 \leq FI(i) \leq c, FI(j) = 0\} \\ \text{IJ}^{[3]} &= \{(i, j) : 1 \leq j < i \leq n, FI(i) = 0, FI(j) = -c\}. \end{aligned}$$

One can easily verify that the union of the first two sets, $\text{IJ}^{[1]} \cup \text{IJ}^{[2]}$, can be mapped bijectively to the edge set \bar{E} , and the third set $\text{IJ}^{[3]}$ can be mapped bijectively to the edge set \tilde{E} . The difference between sets $\text{IJ}^{[1]}$, $\text{IJ}^{[2]}$ and $\text{IJ}^{[3]}$ and \bar{E} and \tilde{E} is that the sets $\text{IJ}^{[1]}$ to $\text{IJ}^{[3]}$ focus on *ordered pairs* while the edges in E correspond to unordered vertex pairs (undirected edges). Also, we can see that there are $\frac{(n-|N_0|)(d-|N_0|)}{2}$ pairs in $\text{IJ}^{[1]}$, $d|N_0|$ pairs in $\text{IJ}^{[2]}$, and $|N_{-c}| \cdot |N_0|$ pairs in $\text{IJ}^{[3]}$. Thus, in total, there are

$$\frac{(n-|N_0|)(d-|N_0|)}{2} + d|N_0| + |N_{-c}| \cdot |N_0| \quad (10)$$

distinct pairs in the overall index set $\text{IJ}^{[1]} \cup \text{IJ}^{[2]} \cup \text{IJ}^{[3]}$. This implies that the total number of edges of graph G is also characterized by (10).

Each edge of graph G corresponds to one coded packet that is stored in the distributed storage system. More specifically, each edge $(i, j) \in \bar{E}$ represents a packet $P_{(i,j)}$ that is stored in the two physical nodes i and j , i.e., both nodes i and j store an identical copy of the packet $P_{(i,j)}$. On the other hand, each edge $(i, j) \in \tilde{E}$ represents a packet $\tilde{P}_{(i,j)}$ that is only stored in one of its two vertices, the corresponding vertex in N_{-c} . One can verify by examining the $\text{IJ}^{[1]}$ to $\text{IJ}^{[3]}$ index sets defined previously that each physical node stores exactly $\alpha = d$ packets.

The outer code: We now describe how to generate the $|\text{IJ}^{[1]}| + |\text{IJ}^{[2]}| + |\text{IJ}^{[3]}|$ coded packets (the $P_{(i,j)}$ and $\tilde{P}_{(i,j)}$ packets depending on whether $(i, j) \in \bar{E}$ or $(i, j) \in \tilde{E}$) from the \mathcal{M} original packets, where \mathcal{M} is specified by (9). Our goal is to design the $|\text{IJ}^{[1]}| + |\text{IJ}^{[2]}| + |\text{IJ}^{[3]}|$ coded packets satisfying the following two properties.

Property 1: For any $i_0 \in N_0$, there are d different j_1 indices satisfying $(j_1, i_0) \in \text{IJ}^{[2]}$ and they are those $j_1 \in N_1 \cup N_2 \cup \dots \cup N_c$ for all $(j_1, i_0) \in \text{IJ}^{[2]}$. We require that any given coded packet $\tilde{P}_{(i_0, j_0)}$ corresponding to some $(i_0, j_0) \in \text{IJ}^{[3]}$ must be a linear combination of the d packets $P_{(j_1, i_0)}$ for

all j_1 satisfying $(j_1, i_0) \in \text{IJ}^{[2]}$. Recall from the inner code description that the packet corresponding to $\tilde{P}_{(i_0, j_0)}$ is stored only in node j_0 . Property 1 means that each of $P_{(i_0, j_0)}$ packets is a linear combination of the d packets stored in node i_0 , those $P_{(j_1, i_0)}$ packets.

We now describe the second required property. One of the main ingredient of this property is to count the edges in $\text{IJ}^{[2]} \cup \text{IJ}^{[3]}$ in a different way to those edges in $\text{IJ}^{[1]}$. Recall that there are $|N_0| = n \bmod (n-d)$ nodes in the incomplete family and they are nodes $c(n-d) + 1$ to $c(n-d) + |N_0|$ where c is the family index of the last complete family. For any subset of the total $|E|$ packets, define a_m , $m = 1$ to $|N_0|$, as the number of packets that correspond to all edges in $E = \bar{E} \cup \tilde{E}$ connected to the m -th vertex $(c(n-d) + m) \in N_0$. Specifically, $\sum_{m=1}^{|N_0|} a_m$ is the number of edges in $\text{IJ}^{[2]} \cup \text{IJ}^{[3]}$. Define a_0 as the number of packets in this subset that correspond to edges that are not connected to any of the vertices in N_0 , those edges in $\text{IJ}^{[1]}$. Define $\text{a.count} \triangleq a_0 + \sum_{m=1}^{|N_0|} \min(a_m, d)$. The above description specifies how to compute a value a.count from any subset of edges.

Property 2: The $|E|$ coded packets must satisfy that any subset of packets having $\text{a.count} \geq \mathcal{M}$ can be used to reconstruct the original \mathcal{M} files.

In the following, we describe how to construct the outer code, i.e., how to design coded packets for the $|E|$ edges that satisfy the above two properties. Specifically, we can use a two-phase approach to generate the packets. We first independently and uniformly randomly generate $|\bar{E}| = \frac{(n-|N_0|)(d-|N_0|)}{2} + d|N_0|$ linearly encoded packets from the \mathcal{M} packets of the original file. These packets are fixed and arbitrarily assigned to the edges in \bar{E} (one for each edge). After this first step, all physical nodes store exactly d packets except those nodes in N_{-c} , each of which now stores exactly $(d - |N_0|)$ packets. Now, from each node in $u \in N_0$, we generate independently and uniformly a random set of $|N_{-c}|$ linearly encoded packets from the d packets stored in u . We fix these newly generated packets and assign them to each of the $|N_{-c}|$ edges in $\{(u, w) \in \tilde{E} : \forall w \in N_{-c}\}$. Specifically, these $|N_{-c}|$ packets will now be stored in node $w \in N_{-c}$, one for each w . Repeat this construction for all $u \in N_0$. After this second step, each edge in $\bar{E} \cup \tilde{E}$ has been assigned one distinct coded packet and each node in $N = N_1 \cup \dots \cup N_c \cup N_{-c} \cup N_0$ now stores exactly d packets. The Phase-1 construction is now complete.

After the initial random-construction phase, we enter the second phase, the verification phase. In this phase, we fix the packets and deterministically check whether they satisfy Property 2 (by our construction the coded packets always satisfy Property 1). The following lemma states that with high probability, the randomly generated packets in Phase 1 will satisfy Property 2.

Lemma 1: When $\text{GF}(q)$ is large enough, with close-to-one probability, the above random construction will satisfy Property 2.

The proof of Lemma 1 is relegated to Appendix A.

Lemma 1 implies that with high-probability, the random construction in Phase 1 will lead to a deterministic set of coded

packets that satisfy Properties 1 and 2. In the rare event that the random construction does not satisfy Property 2, we simply repeat the random construction until we find a set of coded packets that satisfies Properties 1 and 2. Note that this construction is performed off-line during the design stage. Once the coded packets are found by random construction, we will fix the coded packets for future use. It is worth emphasizing that the construction of a code satisfying Properties 1 and 2 is not unique. We may be able to use some other method of construction.³ All our subsequent discussion holds as long as the final coded packets satisfy Properties 1 and 2.

To illustrate the construction/notation of GFR codes, we return to Example 3 of Section IV. In that example, we have $(n, k, d, \alpha, \beta) = (7, 3, 3, 3, 1)$ and $|E| = 12$, $|\bar{E}| = 9$, and $|\tilde{E}| = 3$, see Fig 4. The packets corresponding to the edges in \bar{E} are $P_{(1,5)} = X_1$, $P_{(1,6)} = X_2$, $P_{(1,7)} = X_3$, $P_{(2,5)} = X_4$, $P_{(2,6)} = X_5$, $P_{(2,7)} = X_6$, $P_{(3,5)} = X_7$, $P_{(3,6)} = X_8$, and $P_{(3,7)} = X_9$. On the other hand, the packets corresponding to edges in \tilde{E} are $\tilde{P}_{(5,4)} = X_{10}$, $\tilde{P}_{(6,4)} = X_{11}$, and $\tilde{P}_{(7,4)} = X_{12}$. It is clear by (4) to (6) that the construction satisfies Property 1. The coefficients in (2) and (3) are chosen randomly while using computers to verify that Property 2 is satisfied for the final construction.

Thus far, we have described a new inner/outer code construction that can be used for arbitrary (n, k, d) values. Section VI is dedicated exclusively to proving that such a construction is a legitimate exact-repair RC that achieves the MBR point of the FHS scheme.

VI. ANALYSIS OF THE GFR CODES

A. The Repair Operations

In this section, we first argue that the above GFR code can be exactly-repaired using the FHS scheme. First, consider the case that node i fails for some $i \in N_1 \cup N_2 \cup \dots \cup N_c \cup N_0$ (those in $N \setminus N_{-c}$). The d packets stored in node i thus need to be repaired. We then notice that the d packets in node i correspond to the d edges in \bar{E} that are incident to node i . Therefore, each of those d packets to be repaired is stored in another node j and node i can thus be *repaired-by-transfer*. Note that by our construction, the neighbors of node i are indeed the helper set D_i of the FHS scheme. Also see our discussion in Example 3 of Section IV for illustration.

We now consider the case in which node i in N_{-c} fails. We again notice that $(d - n \bmod (n - d))$ of its d packets correspond to (solid) edges in \bar{E} . Therefore each of those $(d - n \bmod (n - d))$ packets is also stored in another node and can again be *repaired-by-transfer*. To restore the remaining $n \bmod (n - d)$ packets, we notice that by our construction, these packets correspond to the edges in $\{(w, i) \in \tilde{E} : (w, i) \in \text{IJ}^{[3]}\}$. By Property 1 of our outer code construction, for any $w_0 \in N_0$, $\tilde{P}_{(w_0, i)}$, those in $\text{IJ}^{[3]}$, is a linear combination of the d packets $\{P_{(j, w_0)} : (j, w_0) \in \text{IJ}^{[2]}, j = 1, 2, \dots, d\}$ stored

in node w_0 . Thus, during repair, newcomer i can ask physical node w_0 to compute the packet $\tilde{P}_{(w_0, i)}$ and send the final result for all $w_0 \in N_0$. Therefore, newcomer i can exactly-repair all the remaining $n \bmod (n - d)$ packets as well. Also see our discussion in Example 3 of Section IV for illustration.

B. The Reconstruction Operations

The following proposition shows that the GFR code with FHS can protect against any $(n - k)$ simultaneous failures.

Proposition 2: Consider the GFR code with any given (n, k, d) values. For any arbitrary selection of k nodes, one can use all the $k \cdot d$ packets stored in these k nodes (some of them are identical copies of the same coded packets) to reconstruct the original \mathcal{M} file packets.

Since the α , β , and \mathcal{M} values in (9) match the MBR point of the FHS scheme in (8), Proposition 2 shows that the explicitly constructed GFR code indeed achieves the MBR point of the FHS scheme predicted by the min-cut-based analysis.

The rest of this section is dedicated to the proof of Proposition 2.

Proof: Consider an arbitrarily given set of k nodes in the distributed storage network, denoted by S . Denote nodes in S that belong to N_i by $S_i \triangleq S \cap N_i$. We now consider the set of edges that are incident to the given node set S , i.e., those edges have at least one end being in S and each of the edges corresponds to a distinct packet stored in nodes S . Recall that for any set of edges, we can compute the corresponding a.count value as defined in Property 2 of our code construction. The key to the proof is to show that the a.count value of the edges incident to S is no less than \mathcal{M} . Then Property 2 immediately leads to the proof of Proposition 2.

To that end, we describe the following step-by-step procedure, termed COUNT, that computes the value a.count. We will later analyze each step of the procedure to quantify the a.count value.

- 1) We first define $G_1 = (V_1, E_1) = G = (V, E)$ as the original graph representation of the GFR code. Choose an arbitrary order for the vertices in S such that all nodes in S_{-c} come last. Call the i -th vertex in the order, v_i . That is, we have that $S_{-c} = \{v_i : k - |S_{-c}| + 1 \leq i \leq k\}$ and $S_1 \cup \dots \cup S_c \cup S_0 = \{v_i : 1 \leq i \leq k - |S_{-c}|\}$.
- 2) Set $e(S) = 0$, where $e(S)$ will be used to compute a.count.
- Now, do the following step sequentially for $i = 1$ to $|S| = k$:
- 3) Consider vertex v_i . We first compute

$$x_i = |\{(v_i, j) \in E_i \cap \bar{E} : j \in N\}| + 1_{\{v_i \in S_{-c}\}} \cdot \sum_{u \in N_0} 1_{\{(u, v_i) \in E_i \cap \tilde{E}\}} \cdot 1_{\{|\{(u, j) \in E_i : j \in N\}| > |N_{-c}|\}}. \quad (11)$$

Once x_i is computed, update $e(S) = e(S) + x_i$. Remove all the edges incident to v_i from G_i . Denote the new graph by $G_{i+1} = (V_{i+1}, E_{i+1})$.

Intuitively, the above procedure first “counts” the number of edges in G_i that belongs to \bar{E} and is connected to the target

³The computational complexity during the design stage is not the main focus in this work. Therefore, we opted to use the random code construction to demonstrate the existence of a desired code. For practical implementation, some finite-algebra-based construction could drastically reduce the complexity of the construction.

vertex v_i , namely, the $|\{(v_i, j) \in E_i \cap \bar{E} : j \in N\}|$ term in (11). Then, if the target vertex $v_i \in S_{-c}$, we compute one more term in the following way. For each edge $(u, v_i) \in E_i \cap \bar{E}$, if the following inequality holds, we also count this specific (u, v_i) edge:

$$|\{(u, j) \in E_i : j \in N\}| > |N_{-c}|. \quad (12)$$

That is, we check how many edges (including those in $\bar{E} \cap E_i$ and in $\bar{E} \cap E_i$) are still connected to u . We count the single edge (u, v_i) if there are still at least $(|N_{-c}| + 1)$ edges in E_i that are connected to u . Collectively, this additional counting mechanism for the case of $v_i \in S_{-c}$ gives the second term in (11). After counting the edges incident to v_i , we remove those edges from E_i so that in the future counting rounds (rounds $> i$) we do not double count the edges in any way.

Claim 1: After finishing the subroutine COUNT, the final $e(S)$ value is exactly the value of a.count.

Proof: The proof of the above claim is as follows. We first note that in the subroutine, we order the nodes in S in the specific order such that all nodes in S_{-c} are placed last. Therefore, in the beginning of the subroutine COUNT, all the v_i vertices do not belong to S_{-c} . For that reason, the second term in (11) is zero. Since $v_i \notin S_{-c}$, all the edges connected to v_i are in \bar{E} . The first term of (11) thus ensures that we count all those edges in this subroutine. Since we remove those counted edges in each step (from G_i to G_{i+1}), we do not double count any of the edges. Therefore, before we start to encounter a vertex $v_i \in S_{-c}$, the subroutine correctly counts the number of edges incident to the v_j for all $1 \leq j < i$.

We now consider the second half of the subroutine, i.e., when $v_i \in S_{-c}$. We then notice that the subroutine still counts all those edges in \bar{E} through the first term in (11). The only difference between COUNT and a regular counting procedure is the second term in (11). That is, when counting any edge in \bar{E} , we need to first check whether the total number of edges in G_i incident to u is greater than $|N_{-c}|$. To explain why we have this *conditional counting* mechanism, we notice that in the original graph G , each node $u \in N_0$ has $|\{(u, j) \in \bar{E} : j \in N\}| = d$ and $|\{(u, j) \in \bar{E} : j \in N\}| = |N_{-c}|$. Therefore, the total number of edges connected to u is $|\{(u, j) \in E : j \in N\}| = d + |N_{-c}|$. Note that during the counting process, those counted edges are removed from the graph during each step. Since G_i is the remaining graph after removing all those counted edges in the previous $(i - 1)$ steps, if we still have $|\{(u, j) \in E_i : j \in N\}| > |N_{-c}|$, then it means that we have only removed strictly less than $(d + |N_{-c}|) - |N_{-c}| = d$ number of edges in the previous $(i - 1)$ counting rounds. The above argument thus implies that in the previous $(i - 1)$ counting rounds, we have only counted $< d$ edges that are incident to node u .

Without loss of generality, we assume that u is the m -th node of N_0 . Then it means that the a_m value (the number of edges connected to u) computed thus far (until the beginning of the i -th counting round) is still strictly less than d . Therefore, when computing the objective value $\text{a.count} = a_0 + \sum_m \min(a_m, d)$, the to-be-considered edge (v_i, u) in the second term of (11) will increment a_m value

by 1 and thus increment a.count by 1. Since our goal is to correctly compute the a.count value by this subroutine, the subroutine needs to include this edge into the computation, which leads to the second term in (11).

On the other hand, if the total number of edges in G_i that are adjacent to u is $\leq |N_{-c}|$, it means that we have removed $\geq (d + |N_{-c}|) - |N_{-c}| = d$ number of edges in the previous counting rounds. That is, when counting those edges adjacent to u , we have already included/encountered $\geq d$ such edges in the previous $(i - 1)$ rounds. As a result, the corresponding a_m value is $\geq d$. Therefore, when computing the objective value $\text{a.count} = a_0 + \sum_m \min(a_m, d)$, the to-be-considered edge (v_i, u) will increment the value of a_m by 1 but *will not* increment the a.count value. In the subroutine COUNT, we thus do not count the edges in \bar{E}_i anymore, which leads to the second term in (11).

The new constraint put in Step 3 thus ensures that the final output $e(S)$ is the value of a.count. ■

We now need to prove that for any set S of k nodes, the corresponding $e(S) \geq \mathcal{M}$. Assuming this is true, we can then invoke Property 2, which guarantees that we can reconstruct the \mathcal{M} packets of the original file from the coded packets stored in S .

To prove that $e(S) \geq \mathcal{M}$, we need the following claim.

Claim 2: For any arbitrarily given set S , there exists an $\tilde{\mathbf{r}} \triangleq (\tilde{r}_1, \dots, \tilde{r}_k) \in \{1, 2, \dots, n\}^k$ such that

$$e(S) = \sum_{i=1}^k (d - z_i(\tilde{\mathbf{r}})), \quad (13)$$

where $z_i(\cdot)$ is a function $z_i : \{1, \dots, n\}^k \mapsto \mathbb{N}$ defined as $z_i(\mathbf{r}) = |\{a \in D_{r_i} : \exists j < i, a = r_j\}|$, where \mathbb{N} is the set of all positive integers and D_{r_i} is the helper set of node r_i in our GFR code construction. Additional explanation of the function $z_i(\cdot)$ can be found in Part 1 [2, Proposition 2].

Using the above claim, we have

$$\text{a.count} = e(S) = \sum_{i=1}^k (d - z_i(\tilde{\mathbf{r}})) \quad (14)$$

$$\geq \min_{\mathbf{r} \in \{1, \dots, n\}^k} \sum_{i=1}^k (d - z_i(\mathbf{r})) \quad (15)$$

$$= \min_{\pi_f} \sum_{i=1}^k (d - y_i(\pi_f)) \quad (16)$$

$$= \sum_{i=1}^k (d - y_i(\pi_f^*)) \quad (17)$$

$$= \mathcal{M}. \quad (18)$$

where (14) follows from Claim 2; (15) follows from taking the minimum operation; (16) follows from the proof of [2, Proposition 3] where the minimum is taken over all family index permutations π_f and the function $y_i(\pi_f)$ is defined in (8); (17) follows from the optimality of the RFIP in [2, Proposition 7]; and (18) follows from (9). By Property 2, we have thus proved that the $k \cdot d$ packets stored in any set of k nodes can be used to jointly reconstruct the original file of size \mathcal{M} .

The proof of Claim 2 is provided in Appendix B. The proof that the GFR codes can protect against $(n - k)$ simultaneous failures is hence complete. ■

VII. THE EXTENSION OF THE GFR CODES

In our GFR codes, the helper selection scheme is based on the FHS scheme proposed in Part I. In Part I [2], the FHS scheme is also extended to a new scheme called, family-plus scheme. In this section, we introduce some basic notation/concepts of the family-plus helper selection scheme and then discuss how we can generalize the GFR codes in this work so that we can also replace the FHS scheme in the GFR codes by the new family-plus scheme.

A. The Family-Plus Helper Selection Scheme

The family-plus helper selection scheme is an extension of the FHS scheme for $n \gg d$. In family-plus helper selection, the n nodes are grouped into several disjoint groups of $2d$ nodes and one disjoint group of n_{remain} nodes. The first type of groups is termed the regular group while the second type is termed the remaining group. If there has to be one remaining group (when $n \bmod (2d) \neq 0$), then it is enforced that the size of the remaining group is as small as possible but still satisfying $n_{\text{remain}} \geq 2d + 1$. After the partitioning, the FHS scheme is applied to the individual groups. Specifically, if a newcomer belongs to the first group, then all its helpers are chosen within the same group according to the rules of the original FHS scheme. Since each regular group is of size $2d$ nodes and each remaining group must satisfy $n_{\text{remain}} \geq 2d + 1$, one can easily verify that whenever $n \leq 4d$, then there is no regular group and only 1 remaining group. As a result, the family-plus scheme collapses back to the original FHS scheme. On the other hand, when $n \geq 4d + 1$, then there will be multiple groups and the family-plus scheme differs from the FHS scheme.

The file size that can be protected at the MBR point of the family-plus helper selection scheme was found in Part I to be

$$\mathcal{M} = \left(1_{\{n \bmod (2d) \neq 0\}} \cdot \sum_{i=0}^{\min(k, 2d-1)-1} \left(d - i + \left\lfloor \frac{i}{2} \right\rfloor \right) + d^2 \left\lfloor \frac{(k - n_l)^+}{2d} \right\rfloor + \sum_{i=0}^q \left(d - i + \left\lfloor \frac{i}{2} \right\rfloor \right) \right) \beta, \quad (19)$$

where

$$q = ((k - n_l)^+ \bmod (2d)) - 1, \text{ and} \\ n_l = \begin{cases} n_{\text{remain}}, & \text{if } n \bmod (2d) \neq 0 \\ 0, & \text{otherwise.} \end{cases}$$

In Part I, it was proved that for any (n, k, d) values, (19) is always no less than (8). That is, the family-plus scheme improves upon the FHS scheme regardless whether $n \leq 4d - 1$ or $n \geq 4d$.

B. The GFR Codes Based on the Family-Plus Scheme

The GFR codes described above can be modified and used to construct an explicit exact-repair code that can achieve the MBR point of the family-plus helper selection scheme. This is achieved by first applying the same inner code graph construction of the above GFR codes to each group of the family-plus helper selection scheme, i.e., the edge representation of each group consists of the two edge sets \bar{E} and \tilde{E} . Then, since the repair of the family-plus scheme occurs within each group separately, for the outer code, we enforce Property 1 for each individual group so that we can maintain the exact-repair property. Finally, we need to ensure that any subset of k nodes (which could be across multiple groups) can be used to reconstruct the original file. Therefore, we have to ensure that the outer code satisfies a modified version of Property 2.

In the following we briefly describe how to do this modification with a slight abuse of notation. Recall that in the family-plus helper selection scheme, only the remaining group could possibly have an incomplete family. Denote the set of incomplete family nodes in the remaining group by N_0 and the graph of the remaining group by $G_{\text{remain}} = (V_{\text{remain}}, E_{\text{remain}})$. The new property imposed on the packets becomes **Modified Property 2**: Index the vertices in $N_0 \subset V_{\text{remain}}$ by $\{u_1, u_2, \dots, u_{|N_0|}\}$. For any given subset of the total packets (across all groups) and any given m satisfying $1 \leq m \leq |N_0|$, define a_m as the number of packets in this subset that correspond to the edges in $E_{\text{remain}} = \bar{E}_{\text{remain}} \cup \tilde{E}_{\text{remain}}$ that are incident to vertex $u_m \in N_0$. Define a_0 as the number of the other packets in this subset, i.e., those packets not corresponding to any edges that are incident to N_0 . Define $\text{a.count} \triangleq a_0 + \sum_{m=1}^{|N_0|} \min(a_m, d)$. The modified Property 2 enforces that we must be able to reconstruct the original file of size \mathcal{M} if $\text{a.count} \geq \mathcal{M}$.

We can again use the concept of random linear network coding to prove the existence of a code satisfying Property 1 and the Modified Property 2 in a similar way as in Lemma 1. The correctness of the proposed GFR codes for family-plus helper selection schemes can be proved in a similar way as when proving the correctness for FHS schemes provided in Section VI. We omit the detailed proofs since they are simple extensions of the proofs provided for the FHS scheme with only the added notational complexity of handling different groups of nodes in the family-plus helper selection schemes.

VIII. CONCLUSION

In this paper, we have presented a new class of codes that we term *generalized fractional repetition (GFR) codes*. These codes possess several important properties: (i) they achieve the MBR point of the FHS scheme and close the loop of the graph-based necessary and sufficient condition of the benefits of helper selection derived in Part I [2]; (ii) the proposed GFR codes are exact-repair codes and for the most part admit the repair-by-transfer property; and (iii) their construction utilizes a new code-construction technique that generalizes the existing FR codes for arbitrary network parameters. One future direction is to further generalize the proposed GFR codes for

the multiple failures scenario in a way similar to the existing results in [5], [12].

APPENDIX A PROOF OF LEMMA 1

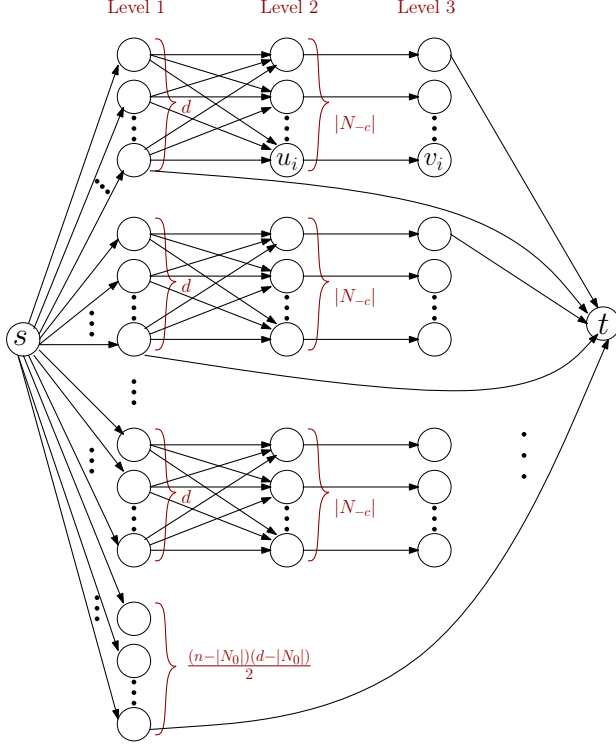


Fig. 6. The graph of the proof of Lemma 1.

To prove this lemma, we model the problem using a finite directed acyclic graph and then we invoke the results from random linear network coding [6]. The graph has a single source vertex s that is incident to $|\bar{E}| = |\mathcal{J}^{[1]}| + |\mathcal{J}^{[2]}| = \frac{(n-|N_0|)(d-|N_0|)}{2} + d|N_0|$ other vertices with edges of capacity 1. We call these vertices *level 1* vertices. Among these level 1 vertices, we focus on a subset of $d|N_0|$ vertices and partition it into $|N_0|$ disjoint groups and each group consists of d arbitrarily chosen distinct vertices. The intuition is that each group of them is associated with a vertex in N_0 . See Fig. 6 for illustration.

Now, in addition to the source s and the level 1 vertices, we add $|N_0| \cdot |N_{-c}|$ new node pairs (u_i, v_i) for all $1 \leq i \leq |N_0| \cdot |N_{-c}|$. Each (u_i, v_i) is connected by an edge of capacity 1. We call the u_i nodes, level 2 vertices and the v_i nodes level 3 vertices. We partition the new node pairs (edges) into $|N_0|$ groups and each group consists of $|N_{-c}|$ edges. We then associate each group of $|N_{-c}|$ edges to one group of d level 1 vertices created previously. See Fig. 6 for illustration. Finally, for the level 1, level 2, and level 3 vertices belonging to the same group (there are $|N_0|$ groups in total), we connect all the level 1 vertices in this group and all the level 2 vertices in this group by an edge with infinite capacity.

We now describe the relationship of the newly constructed graph in Fig. 6 to the graph representation of the GFR code. For easier reference, we use the graph in Fig. 6 to refer to the

newly constructed graph; and use the graph in Fig. 4 to refer to the graph representation of the GFR codes. There are $|N_0|$ groups in the graph of Fig. 6 and each group corresponds to one node in N_0 of the graph of Fig. 4. We notice that there are $|\bar{E}| = \frac{(n-|N_0|)(d-|N_0|)}{2} + d|N_0|$ number of level 1 vertices in the graph of Fig. 6 and $|\bar{E}| = \frac{(n-|N_0|)(d-|N_0|)}{2} + d|N_0|$ number of edges in \bar{E} of the graph of Fig. 4. As a result, we map each level 1 vertex bijectively to an edge in \bar{E} in a way that each group of the level-1 vertices in Fig. 6 (totally $|N_0|$ groups) corresponds to the d edges in \bar{E} that are connected to the same node in N_0 of Fig. 4.

There are $|N_0| \cdot |N_{-c}|$ number of level 3 vertices in the graph of Fig. 6 and there are $|N_0| \cdot |N_{-c}|$ number of \bar{E} edges in the graph of Fig. 4. As a result, we map each level 3 vertex bijectively to an edge in \bar{E} in a way that each group of the level-3 vertices in Fig. 6 (totally $|N_0|$ groups) corresponds to the $|N_{-c}|$ edges in \bar{E} that are connected to the corresponding node in N_0 of Fig. 4.

We now focus on how to encode over the graph of Fig. 6 and then use the above mapping to convert it to a coding scheme over the graph of Fig. 4. Assume that source s has a file of \mathcal{M} packets. We perform random linear network coding (RLNC) [6] on the graph of Fig. 6 assuming a sufficiently large finite field $\text{GF}(q)$ is used. Specifically, for any level-1 vertex u , the corresponding (s, u) is a random mixture of all \mathcal{M} packets. For each (u_i, v_i) edge connecting a level-2 vertex u_i and a level-3 vertex v_i , it carries a linear combination of all coded (s, u) edges that are incident⁴ to (u_i, v_i) . After the encoding over Fig. 6 is fixed, we can immediately construct the corresponding encoding scheme over Fig. 4 based on the aforementioned mapping. For example, for a level-2 vertex u and a level-3 vertex v , if edge (u, v) belongs to the i_0 -th group in Fig. 6 and v is the j_0 -th level 3 vertex in this group, then, we assign the coded packets on the edge (u, v) to the edge $e \in \bar{E}$ (in the graph of Fig. 4) that connects the i_0 -th node in N_0 and the j_0 -th node in N_{-c} .

In the following, we will prove that with a sufficiently large $\text{GF}(q)$ the above code construction (from the RLNC-based code in the graph of Fig. 6 to the GFR codes in the graph of Fig. 4) satisfies Properties 1 and 2 with close-to-one probability.

We first prove that our construction satisfies Property 1 with probability one. To that end, we notice that any coded packet $\tilde{P}_{(i_0, j_0)}$ corresponding to some $(i_0, j_0) \in \mathcal{I}^{[3]}$ in the graph of Fig. 4 is now mapped from a (u, v) edge in Fig. 6 where u is a level 2 vertex; v is a level 3 vertex; (u, v) belongs to the i_0 -th group in Fig. 6; and v is the j_0 -th level 3 vertex in this group. By the graph construction in Fig. 6, such a coded packet is a linear combination of the coded packets in Fig. 6 from source s to vertex \tilde{u} where the \tilde{u} vertices are the level-1 vertices corresponding to the i_0 -th group. Since those packets along (s, \tilde{u}) are the $P_{(j_1, i_0)}$ packets for all j_1 satisfying $(j_1, i_0) \in \mathcal{I}^{[2]}$ in the graph of Fig. 4, we have thus proved Property 1: Specifically, any coded packet $\tilde{P}_{(i_0, j_0)}$ corresponding to some

⁴Technically, we should say all (s, u) edges that are *upstream* of (u_i, v_i) . However, since the edges connecting level-1 and level-2 vertices are of infinite capacity, we use the word *incident* in a loose sense.

$(i_0, j_0) \in \text{IJ}^{[3]}$ is a linear combination of the packets $P_{(j_1, i_0)}$ for all j_1 satisfying $(j_1, i_0) \in \text{IJ}^{[2]}$.

To prove that the above construction satisfies Property 2 with close-to-one probability, for any edge set subset of edges in the graph of Fig. 4 with the corresponding a.count value satisfying $\text{a.count} \geq \mathcal{M}$, we place a sink node t in the graph of Fig. 6 that connects to the corresponding set of level 1/level 3 vertices in Fig. 6 using edges of infinite capacity. See Fig. 6 for illustration of one such t . One can quickly verify that the min-cut-value from the source s to the sink t in the graph of Fig. 6 is indeed the a.count value computed from the given subset of edges in the graph of Fig. 4. As a result, with a sufficiently large finite field $\text{GF}(q)$, sink t in Fig. 6 can successfully reconstruct the original file with close-to-one probability. Since the sink t accesses only level 1 and level 3 vertices, the $P_{(i,j)}$ packets in the graph of Fig. 4 that correspond to the level 1 vertices in the graph of Fig. 6 and the $\tilde{P}_{(i,j)}$ packets in the graph of Fig. 4 that correspond to the level 3 vertices in the graph of Fig. 6 jointly can reconstruct the original file of size \mathcal{M} . Property 2 is thus also satisfied. Since, there are at most $\binom{|E|}{\mathcal{M}}$ different ways of choosing the sink t ,⁵ a very loose outer bound of the success probability is

$$\Pr(\text{The RLNC construction satisfies Lemma 1}) \geq 1 - \frac{\binom{|E|}{\mathcal{M}}}{q} \quad (20)$$

By the above arguments, the proof of Lemma 1 is complete.

APPENDIX B PROOF OF CLAIM 2

In order to prove Claim 2, we will need the following fact.

Claim 3: Suppose there exists a node $a \in S_{-c}$ and a node $b \in N_c \setminus S_c$. Define a new set of nodes $S' \triangleq (S \cup \{b\}) \setminus a$. That is, we remove node a from S but add a new node b in S that satisfies $b \in N_c$. Then

$$e(S) = e(S'). \quad (21)$$

That is, running the subroutine COUNT on both S and S' will lead to the same final output value.

Proof: We consider COUNT for the set S' and we denote nodes in S' that belong to N_i by $S'_i \triangleq S' \cap N_i$. To avoid confusion when S' is used as input to the subroutine COUNT, we call the new graphs during the counting steps of COUNT by $G'_i = (V'_i, E'_i)$, the new vertices by v'_i , and the new x_i by x'_i . Since the subroutine COUNT can be based on any sorting order of nodes in S (and in S') as long as those nodes in N_{-c} come last, we assume that the nodes in S are sorted in a way that node a is the very first node in S_{-c} . For convenience, we say that node a is the i_0 -th node in S and we assume that all the first $(i_0 - 1)$ -th nodes are not in S_{-c} and all the nodes following the $(i_0 - 1)$ -th node are in S_{-c} . That is, $i_0 = |S| - |S_{-c}| + 1 = k + 1 - |S_{-c}|$. We now use the same sorting order of S and apply it to S' . Specifically, the i -th node of S

is the same as the i -th node in S' except for the case of $i = i_0$. The i_0 -th node of S' is set to be node b . One can easily check that the sorting orders of S and S' both satisfy the required condition in Step 1 of the subroutine COUNT.

We will run COUNT on both S and $(S \cup \{b\}) \setminus a$ in parallel and compare the resulting $e(S)$ and $e((S \cup \{b\}) \setminus a)$.

It is clear that in rounds 1 to $(i_0 - 1)$, the subroutine COUNT behaves identically when applied to the two different sets S and $S' = (S \cup \{b\}) \setminus a$ since their first $(i_0 - 1)$ vertices are identical. We now consider the i_0 -th round and argue that the total number of edges in E'_{i_0} incident to v'_{i_0} is equal to the total number of edges incident to v_{i_0} in E_{i_0} . Recall that b and a have the same helper sets since they are from the same complete family. Specifically, the edges in E incident to $v_{i_0} = a \in S_{-c}$ that have been counted in the first $(i_0 - 1)$ rounds are of the form (u, a) for all $u \in \{v_1, v_2, \dots, v_{i_0-1}\} \cap (S_0 \cup S_1 \cup \dots \cup S_{c-1})$. Also note that in the original graph G , there are exactly d edges incident to node $a \in S_{-c}$ (some of them are in \bar{E} and some of them in \tilde{E}). Therefore, in E_{i_0} (after removing those previously counted edges), there are $(d - |\{v_1, v_2, \dots, v_{i_0-1}\} \cap (S_0 \cup S_1 \cup \dots \cup S_{c-1})|)$ number of edges that are incident to v_{i_0} .

Similarly, the edges in E'_{i_0} incident to $v'_{i_0} = b \in S'_c$ that have been counted previously are of the form (u, b) for all $u \in \{v_1, v_2, \dots, v_{i_0-1}\} \cap (S_0 \cup S_1 \cup \dots \cup S_{c-1})$ since $v'_i = v_i$ for $1 \leq i \leq i_0 - 1$ and $S'_x = S_x$ for $0 \leq x \leq c - 1$. Also note that, in the original graph G' , there are exactly d edges incident to node $b \in S'_c$ (all of them are in \bar{E}'). Therefore, in E'_{i_0} (after removing those previously counted edges), there are $(d - |\{v_1, v_2, \dots, v_{i_0-1}\} \cap (S_0 \cup S_1 \cup \dots \cup S_{c-1})|)$ number of edges that are incident to $v'_{i_0} = b$.

We now argue that all the edges in E_{i_0} that are incident to a will contribute to the computation of x_{i_0} . The reason is that node a is the first vertex in S_{-c} . Therefore, when in the i_0 -th counting round, no edge of the form (u, v) where $u \in N_0 \setminus S_0$ and $v \in N_{-c}$ has ever been counted in the previous $(i_0 - 1)$ rounds. Also, since we choose $b \in N_c \setminus S$ to begin with, when running COUNT on S , for all $u \in N_0 \setminus S_0$ at least one edge, edge (u, b) , is not counted during the first $(i_0 - 1)$ rounds. As a result, for any $u \in N_0 \setminus S_0$, in the i_0 -th round, at least $|\{(u, v) : v \in N_{-c}\}| + 1 = |N_{-c}| + 1$ edges incident to u are still in E_{i_0} (not removed in the previous $(i_0 - 1)$ rounds). This thus implies that the second term of (11) will be non-zero. Therefore, at the i_0 -th iteration of Step 3 of COUNT, all the edges in E_{i_0} incident to $v_{i_0} = a$ are counted. The x_{i_0} value computed in (11) thus becomes $x_{i_0} = d - |\{v_1, v_2, \dots, v_{i_0-1}\} \cap (S_0 \cup S_1 \cup \dots \cup S_{c-1})|$.

The previous paragraph focuses on the i_0 -th round when running the subroutine COUNT on S . We now consider the i_0 -th round when running COUNT on S' . We argue that all the edges in E'_{i_0} that are incident to b will contribute to the computation of x'_{i_0} . The reason is that node $b \in S'_c$. Therefore, all edges incident to b belong to \bar{E}' . As a result, all the edges in E'_{i_0} that are incident to b will contribute to the computation of x'_{i_0} through the first term in (11). We thus have $x'_{i_0} = d - |\{v_1, v_2, \dots, v_{i_0-1}\} \cap (S_0 \cup S_1 \cup \dots \cup S_{c-1})|$.

Since $x_{i_0} = x'_{i_0}$, we thus have $e(S) = e(S')$ after the first i_0 counting rounds.

⁵Since ultimately we are only interested in reconstructing the file from any k nodes, we actually only need to consider $\binom{n}{k}$ ways of choosing the sink t , which can further improve the probability lower bound.

We now consider rounds $(i_0 + 1)$ to k . We observe that by our construction $v'_i = v_i \in S'_{-c} \subset S_{-c}$ for $i_0 + 1 \leq i \leq k$. Moreover, since $v_{i_0} = a \in S_{-c}$ and $v'_{i_0} = b \in S'_c$, both vertices a and b are initially not connected to any vertices in S_{-c} and S'_{-c} respectively (those v_i and v'_i with $i_0 + 1 \leq i \leq k$) since vertices of the same family are not connected. Therefore, replacing the i_0 -th node $v_{i_0} = a$ by $v'_{i_0} = b$ will not change the value of the first term in (11) when computing x_i for the i -th round where $i_0 + 1 \leq i \leq k$.

We now consider the second term of (11). For any $u \in S_0$, any edge incident to u has been counted in the first $(i_0 - 1)$ rounds since we assume that when we are running COUNT on the S set, we examine the nodes in S_{-c} in the very last. Therefore, there is no edge of the form (v_i, u) in E_i (resp. $(v'_i, u) \in E'_i$) with $u \in S_0$ since those edges have been removed previously. Therefore, the summation over $u \in N_0$ can be replaced by $u \in N_0 \setminus S_0$ during the i_0 -th round to the k -th round. On the other hand, for any $u \in N_0 \setminus S_0$, if there is an edge connecting $(a, u) \in \bar{E}$, then by our construction there is an edge $(b, u) \in \bar{E}$. Therefore, in the i_0 -th round, the same number of edges incident to u is removed regardless whether we are using S as the input to the subroutine COUNT or we are using S' as the input to the subroutine COUNT. As a result, in the beginning of the $(i_0 + 1)$ -th round, for any $u \in N_0$, we have the following equality

$$|\{(u, j) \in E_i : j \in N\}| = |\{(u, j) \in E'_i : j \in N\}| \quad (22)$$

when $i = i_0 + 1$. Moreover, for any $u \in N_0 \setminus S_0$, we remove one and only one edge (u, v_i) in the i -th round, regardless whether we are counting over S or over S' . Since $v_i = v'_i$ for all $i = i_0 + 1$ to k , we have (22) for all $i = i_0 + 1$ to k as well. The above arguments thus prove that the second term of (11) does not change regardless whether we count over S or S' . As a result, $x'_i = x_i$ for $i_0 + 1 \leq i \leq k$. Since $e(S) = e(S')$ for all k rounds of the counting process, we have thus proved (21). ■

We now turn our attention back to proving Claim 2. For any node set S , by iteratively using Claim 3, we can construct another node set S' such that $e(S) = e(S')$ while either (Case i) $S'_{-c} = \emptyset$; or (Case ii) $S'_{-c} \neq \emptyset$ and $S'_c = N_c$. As a result, we can assume without loss of generality that we have either (Case i) $S_{-c} = \emptyset$; or (Case ii) $S_{-c} \neq \emptyset$ and $S_c = N_c$ to begin with.

We first consider the former case. Let $\tilde{\mathbf{r}}$ be any vector in R such that its $\tilde{r}_i = v_i$ for $1 \leq i \leq k$, i.e., \tilde{r}_i equals the node index of the vertex v_i . We will run the subroutine COUNT sequentially for $i = 1$ to k and compare the increment of $e(S)$ in each round, denoted by x_i in (11), to the i -th term $(d - z_i(\tilde{\mathbf{r}}))$ in the summation of the right-hand side of (13). Consider the i -th round of counting for some $1 \leq i \leq k$, and assume that the corresponding vertex v_i belongs to the y -th family, i.e., $v_i \in N_y$. Since $S_{-c} = \emptyset$ in this case, we have $v_i \notin S_{-c}$ and the second term in (11) is always 0. Therefore, the procedure COUNT is indeed counting the number of edges in \bar{E} that are incident to S without the special conditional counting mechanism in the second term of (11). Therefore,

we have

$$\begin{aligned} x_i &= |\{(v_i, j) \in E_i \cap \bar{E} : j \in N\}| \\ &= d - |\{v_j \notin N_y : v_j \in S, 1 \leq j \leq i - 1\}|, \end{aligned} \quad (23)$$

where d is the number of \bar{E} edges in the original graph G that are incident to v_i and $|\{v_j \notin N_y : v_j \in S, 1 \leq j \leq i - 1\}|$ is the number of edges removed during the first $(i - 1)$ counting rounds. On the other hand, we have

$$v_j \in D_y \Leftrightarrow v_j \in D_y \setminus N_{-c} \Leftrightarrow v_j \in N \setminus (N_y \cup N_{-c}) \quad (24)$$

where the first equality follows from that $S_{-c} = \emptyset$ implies $v_j \notin N_{-c}$; the second equality follows from the FHS construction that $D_y \setminus N_{-c} = N \setminus (N_{-c} \cup N_y)$ for any family index $y \neq -c$. By the definition of function $z_i(\cdot)$, our construction of $\tilde{\mathbf{r}}$ thus always has $|\{v_j \notin N_y : v_j \in S, 1 \leq j \leq i - 1\}| = z_i(\tilde{\mathbf{r}})$. As a result, $x_i = (d - z_i(\tilde{\mathbf{r}}))$ for $i = 1$ to k and our explicitly constructed vector $\tilde{\mathbf{r}}$ satisfies (13).

We now turn our attention to the second case when $S_{-c} \neq \emptyset$ and $S_c = N_c$. Recall that there are k nodes in the set S . Let \mathbf{r} be any vector in R such that its $r_i = v_i$ for $1 \leq i \leq k$. Define j^* as the value that simultaneously satisfies (i) $k - |S_{-c}| \leq j^* \leq k$ and (ii) there are exactly d entries in the first j^* coordinates of \mathbf{r} that are in $N \setminus N_0$. If no value satisfies the above two conditions simultaneously, set $j^* = k + 1$. We now construct another vector $\tilde{\mathbf{r}}$ from \mathbf{r} as follows: Replace the values of the $(j^* + 1)$ -th coordinate to the k -th coordinate of \mathbf{r} by n , the node index of the last node in N_0 and denote the final vector by $\tilde{\mathbf{r}}$.

We will now prove that the above explicit construction of $\tilde{\mathbf{r}}$ satisfies the desired property in (13). The proof is divided into two cases:

Case 1: There exists such a j^* satisfying (i) and (ii). We will run the subroutine COUNT again and compare x_i to the i -th term $(d - z_i(\tilde{\mathbf{r}}))$.

We then observe the following facts:

- 1) In COUNT, from $i = 1$ to $(k - |S_{-c}|)$. For any i in this range, we must have $FI(v_i) \neq -c$, i.e., the family index of node v_i is not $-c$, since we run the subroutine COUNT using a specific ordering of the nodes in S , which examines the nodes in S_{-c} in the very last. As a result, the second term of (11) is always zero. Therefore (23) still holds. By the definition of function $z_i(\cdot)$, our construction of $\tilde{\mathbf{r}}$, and the fact that $1 \leq i \leq k - |S_{-c}|$ (implying no $v_j \in S_{-c}$ for all $1 \leq j \leq i - 1$), we get $x_i = d - z_i(\tilde{\mathbf{r}})$ for all $1 \leq i \leq k - |S_{-c}|$.
- 2) We now consider the case of $i = k - |S_{-c}| + 1$ to j^* of Step 3. For any i in this range, we have $v_i \in S_{-c}$. We now argue that $|\{(u, j) \in E_i : j \in N\}| > |N_{-c}|$ for all edges $(u, v_i) \in E_i \cap \bar{E}$ satisfying $u \in N_0$. The reason is that $(u, v_i) \in E_i$ implies that node u is not counted in the previous $(i - 1)$ rounds, i.e., $u \neq v_{i'}$ for all $1 \leq i' \leq i - 1$. Therefore, an edge of (u, v) is removed if and only if there is a $v = v_j$ for some v_j that is not in N_0 . Since there are exactly d vertices in $\{v_1, v_2, \dots, v_{j^*}\}$ that are not in N_0 , it means that the first $(i - 1)$ counting rounds where $1 \leq i \leq j^*$ can remove at most $(d - 1)$ edges incident to such a node u . Since node u has $(d + |N_{-c}|)$ number of

incident edges in the original graph G , we know that the inequality $|\{(u, j) \in E_i : j \in N\}| > |N_{-c}|$ must hold in the i -th round. As a result, the second term of (11) is non-zero when $i = k - |S_{-c}| + 1$ to j^* and we can thus rewrite

$$x_i = |\{(v_i, j) \in E_i : j \in N\}| \\ = d - |\{v_j \notin N_c \cup N_{-c} : v_j \in S, 1 \leq j \leq i-1\}|.$$

By the definition of function $z_i(\cdot)$ and our construction of $\tilde{\mathbf{r}}$, we get $x_i = d - z_i(\tilde{\mathbf{r}})$ for all $k - |S_{-c}| + 1 \leq i \leq j^*$.

- 3) We now consider the $(j^* + 1)$ -th to the k -th round of Step 3. We claim that

$$x_i = d - |S_1 \cup S_2 \cup \dots \cup S_c| \quad (25)$$

for those $j^* + 1 \leq i \leq k$. The reason behind this is the following. Since $j^* + 1 \leq i \leq k$, we have $v_i \in S_{-c}$. For any $u \in N_0 \setminus S_0$ (those $u \in S_0$ have been considered in the first $(k - |S_{-c}|)$ rounds), there are $(d + |N_{-c}|)$ number edges incident to u in the original graph G . On the other hand, since $i \geq j^* + 1$ and by our construction, there are d entries in the first j^* coordinates of $\tilde{\mathbf{r}}$ that are not in N_0 , we must have removed at least d edges incident to u during the first $(i - 1)$ counting rounds as discussed in the previous paragraph. Therefore, the number of incident edges in E_i that are incident to $u \in N_0 \setminus S_0$ must be $\leq |N_{-c}|$. The second term of (11) is thus zero. As a result, the x_i computed for v_i will only include those edges in $E_i \cap \bar{E}$ incident to it. Since any $v_i \in S_{-c}$ only has $(d - |N_0|)$ number of edges in \bar{E} to begin with, we have that

$$x_i = (d - |N_0|) - |S_1 \cup S_2 \cup \dots \cup S_{c-1}|$$

where $|S_1 \cup S_2 \cup \dots \cup S_{c-1}|$ is the number of edges in \bar{E} that have been removed during the first $(i - 1)$ rounds. Since $S_c = N_c$ in the scenario we are considering and since $|N_c| = |N_0| = n \bmod (n - d)$ in the FHS scheme, we can consequently rewrite x_i as

$$x_i = d - |S_1 \cup S_2 \cup \dots \cup S_c|$$

for $(j^* + 1) \leq i \leq k$. Recall that in the newly constructed $\tilde{\mathbf{r}}$, the values of the $(j^* + 1)$ -th coordinate to the k -th coordinate are n , which belongs to N_0 . Thus, by the definition of function $z_i(\cdot)$, we can see that each of these coordinates only contributes

$$z_i(\tilde{\mathbf{r}}) = |\{\tilde{r}_j \in N \setminus (N_{-c} \cup N_0) : 1 \leq j \leq i-1\}| \\ = |\{\tilde{r}_j \in N \setminus (N_{-c} \cup N_0) : 1 \leq j \leq j^*\}| \quad (26) \\ = |S_1 \cup S_2 \cup \dots \cup S_c|$$

where (26) follows from the fact that in the construction of $\tilde{\mathbf{r}}$, the $(j^* + 1)$ -th to the k -th coordinates of $\tilde{\mathbf{r}}$ are always of value $n \in N_0$. Hence, we get $x_i = d - z_i(\tilde{\mathbf{r}})$ for $(j^* + 1) \leq i \leq k$.

We have proved for this case that $x_i = d - z_i(\tilde{\mathbf{r}})$ for $i = 1$ to k . Therefore, we get (13).

Case 2: No such j^* exists. This means that one of the following two sub-cases is true. Case 2.1: even when choosing

the largest $j^* = k$, we have strictly less than d entries that are not in N_0 . Case 2.2: Even when choosing the smallest $j^* = k - |S_{-c}|$, we have strictly more than d entries that are not in N_0 .

Case 2.1 can be proved by the same arguments used in the previous proof of Case 1 (when proving the scenario of $k - |S_{-c}| + 1 \leq i \leq j^*$), which implies that we have $x_i = d - z_i(\tilde{\mathbf{r}})$ for all $1 \leq i \leq k$. The proof of this case is complete.

Case 2.2 is actually an impossible case. The reason is that for any $1 \leq i \leq k - |S_{-c}|$, there are exactly $|S_1| + |S_2| + \dots + |S_c|$ nodes v_i that are not in N_0 , and we also have

$$\sum_{m=1}^c |S_m| \leq \sum_{m=1}^c |N_m| = d,$$

where the equality follows from our FHS construction. This, together with the observation that the first $(k - |S_{-c}|)$ coordinates of \mathbf{r} are transcribed from the distinct nodes in $S_1 \cup S_2 \cup \dots \cup S_c$, implies that we cannot have strictly more than d entries that are not in N_0 in the first $(k - |S_{-c}|)$ coordinates of \mathbf{r} . Case 2.2 is thus an impossible case.

By the above arguments, the proof of Claim 2 is complete.

REFERENCES

- [1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.
- [2] I. Ahmad and C.-C. Wang, "When can helper node selection improve regenerating codes? Part I: Graph-based analysis," *this issue*, vol. 0, no. 0, pp. 0–0, 2015.
- [3] V. R. Cadambe, S. A. Jafar, H. Maleki, K. Ramchandran, and C. Suh, "Asymptotic interference alignment for optimal repair of mds codes in distributed storage," *IEEE Trans. Inf. Theory*, vol. 59, no. 5, pp. 2974–2987, 2013.
- [4] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4539–4551, 2010.
- [5] S. El Rouayheb and K. Ramchandran, "Fractional repetition codes for repair in distributed storage systems," in *Proc. 48th Annual Allerton Conf. on Comm., Contr., and Computing*, Monticello, IL, Sep. 2010, pp. 1510–1517.
- [6] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Trans. Inf. Theory*, vol. 52, no. 10, pp. 4413–4430, 2006.
- [7] G. M. Kamath, N. Silberstein, N. Prakash, A. S. Rawat, V. Lalitha, O. O. Koyluoglu, P. Kumar, and S. Vishwanath, "Explicit mbr all-symbol locality codes," in *Proc. IEEE Int. Symp. Information Theory (ISIT)*, Istanbul, Turkey, Jul. 2013, pp. 504–508.
- [8] J. H. Kim and V. H. Vu, "Generating random regular graphs," in *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, San Diego, CA, Jun. 2003, pp. 213–222.
- [9] J. C. Koo and J. T. Gill, "Scalable constructions of fractional repetition codes in distributed storage systems," in *Proc. 49th Annu. Allerton Conf. Communication, Control, and Computing*, Monticello, IL, Sep. 2011, pp. 1366–1373.
- [10] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inf. Theory*, vol. 49, no. 2, pp. 371–381, 2003.
- [11] O. Olmez and A. Ramamoorthy, "Repairable replication-based storage systems using resolvable designs," in *Proc. 50th Annu. Allerton Conf. Communication, Control, and Computing*, Monticello, IL, Sep. 2012, pp. 1174–1181.
- [12] —, "Replication based storage systems with local repair," in *International Symposium on Network Coding (NetCod)*, Calgary, AB, Jun. 2013, pp. 1–6.
- [13] S. Pawar, N. Noorshams, S. El Rouayheb, and K. Ramchandran, "Dress codes for the storage cloud: Simple randomized constructions," in *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*, Jul. 2011, pp. 2338–2342.

- [14] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Optimal exact-regenerating codes for distributed storage at the msr and mbr points via a product-matrix construction," *IEEE Trans. Inf. Theory*, vol. 57, no. 8, pp. 5227–5239, 2011.
- [15] K. V. Rashmi, N. B. Shah, P. V. Kumar, and K. Ramchandran, "Explicit construction of optimal exact regenerating codes for distributed storage," in *Proc. 47th Annu. Allerton Conf. Communication, Control, and Computing*, Monticello, IL, Sep. 2009, pp. 1366–1373.
- [16] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, "Distributed storage codes with repair-by-transfer and nonachievability of interior points on the storage-bandwidth tradeoff," *IEEE Trans. Inf. Theory*, vol. 58, no. 3, pp. 1837–1852, 2012.
- [17] —, "Interference alignment in regenerating codes for distributed storage: Necessity and code constructions," *IEEE Trans. Inf. Theory*, vol. 58, no. 4, pp. 2134–2158, 2012.
- [18] Y. Wu, "Existence and construction of capacity-achieving network codes for distributed storage," *IEEE J. Select. Areas Commun.*, vol. 28, no. 2, pp. 277–288, 2010.
- [19] —, "A construction of systematic mds codes with minimum repair bandwidth," *IEEE Trans. Inf. Theory*, vol. 57, no. 6, pp. 3738–3741, 2011.
- [20] Y. Wu and A. G. Dimakis, "Reducing repair traffic for erasure coding-based storage via interference alignment," in *Proc. IEEE Int. Symp. Information Theory (ISIT)*, Seoul, South Korea, Jul. 2009, pp. 2276–2280.
- [21] B. Zhu, K. Shum, H. Li, and H. Hou, "General fractional repetition codes for distributed storage systems," *IEEE Communications Letters*, vol. 18, no. 4, pp. 660–663, 2014.